```
// This source code is subject to the terms of the Mozilla Public License 2.0 at https://mozilla.org/M
// © HeWhoMustNotBeNamed

//@version=4
study("Double Top/Bottom", overlay=true, max_bars_back=300)
pvtLenL = input(6)
pvtLenR = input(4)
waitforclose = input(true)
filterPivots = input(true)
maxPivotArraySize = 5

checkForAbsolutePeaks = input(true)
absolutePeakLoopback = input(20, step=10)

considerMovingAverage = input(true)
MAType = input(title="Moving Average Type", defval="hma", options=["ema", "sma", "hma", "rma", "vwma",
MALengthFast = input(20, step=10)
MALengthSlow = input(100, step=50)

considerPivotDistance = input(true)
atrLength = input(22)
maxAtrDistanceBase = input(2)
maxAtrDistanceHighLow = input(4)

pivotDisplay = input(title="Pivot Display mode", defval="Actual", options=["Actual", "Filtered", "None
displayMovingAverage = input(true)
colorCandles = input(true)

showActualPivots = pivotDisplay == "Actual"
showFilteredPivots = pivotDisplay == "Filtered"

var ph_positions = array.new_int(maxPivotArraySize, 0)
var ph_vals = array.new_float(maxPivotArraySize, 0.0)
var ph_highlow = array.new_bool(maxPivotArraySize, false)

var pl_positions = array.new_int(maxPivotArraySize, 0)
var pl_vals = array.new_float(maxPivotArraySize, 0.0)
var pl_highlow = array.new_bool(maxPivotArraySize, false)

f_getMovingAverage(source, MAType, length)=>
    ma = (MAType == "sma")? sma(source,length):
              (MAType == "ema")? ema(source,length):
              (MAType == "hma")? hma(source,length):
              (MAType == "rma")? rma(source,length):
              (MAType == "vwma")? vwma(source,length):
              (MAType == "wma")? wma(source,length):
              (highest(length)+lowest(length))/2
    ma

f_calculatePivots(pvtLenL, pvtLenR, waitforclose)=>
    Shunt = waitforclose? 1: 0
    pvthi_ = pivothigh(high, pvtLenL, pvtLenR)

```

```
    pvtlo_ = pivotlow(low, pvtLenL, pvtLenR)

    pvthi = pvthi_[Shunt]
    pvtlo = pvtlo_[Shunt]

    higherhigh = na(pvthi) ? na : ( valuewhen(pvthi, high[pvtLenR+Shunt], 1) < valuewhen(pvthi, high[p
    lowerhigh  = na(pvthi) ? na : ( valuewhen(pvthi, high[pvtLenR+Shunt], 1) > valuewhen(pvthi, high[p
    higherlow  = na(pvtlo) ? na : ( valuewhen(pvtlo, low[pvtLenR+Shunt], 1)  < valuewhen(pvtlo, low[pv
    lowerlow   = na(pvtlo) ? na : ( valuewhen(pvtlo, low[pvtLenR+Shunt], 1)  > valuewhen(pvtlo, low[pv

    if(not na(higherhigh) or not na(lowerhigh))
        addPivotHigh = true
        if(array.get(ph_positions, 0) > array.get(pl_positions, 0)) and filterPivots
            if pvthi > array.get(ph_vals, 0)
                array.shift(ph_positions)
                array.shift(ph_vals)
                array.shift(ph_highlow)
            else
                addPivotHigh := false

        if(addPivotHigh)
            array.unshift(ph_positions, bar_index)
            array.unshift(ph_vals, pvthi)
            array.unshift(ph_highlow,na(lowerhigh))

    if(not na(higherlow) or not na(lowerlow))
        addPivotLow = true
        if(array.get(pl_positions, 0) > array.get(ph_positions, 0)) and filterPivots
            if pvtlo < array.get(pl_vals, 0)
                array.shift(pl_positions)
                array.shift(pl_vals)
                array.shift(pl_highlow)
            else
                addPivotLow := false
        if(addPivotLow)
            array.unshift(pl_positions, bar_index)
            array.unshift(pl_vals, pvtlo)
            array.unshift(pl_highlow,na(lowerlow))

    [higherhigh, lowerhigh, higherlow, lowerlow]


[higherhigh, lowerhigh, higherlow, lowerlow] = f_calculatePivots(pvtLenL, pvtLenR, waitforclose)

Shunt = waitforclose? 1: 0
plotshape(showActualPivots ? higherhigh : na, title='HH', style=shape.triangleup,   location=location.
plotshape(showActualPivots ? higherlow  : na, title='HL', style=shape.triangledown,   location=locatic
plotshape(showActualPivots ? lowerhigh  : na, title='LH', style=shape.triangleup, location=location.at
plotshape(showActualPivots ? lowerlow   : na, title='LL', style=shape.triangledown, location=location.

lastHigh = array.get(ph_vals, 0)
llastHigh = array.get(ph_vals, 1)
lastLow = array.get(pl_vals, 0)
llastLow = array.get(pl_vals, 1)
```

```
lastHighBar = bar_index - array.get(ph_positions, 0) + pvtLenR + Shunt
lastLowBar = bar_index - array.get(pl_positions, 0) + pvtLenR + Shunt

llastHighBar = bar_index - array.get(ph_positions, 1) + pvtLenR + Shunt
llastLowBar = bar_index - array.get(pl_positions, 1) + pvtLenR + Shunt

mafast = f_getMovingAverage(close, MAType, MALengthFast)
maslow = f_getMovingAverage(close, MAType, MALengthSlow)

plot(considerMovingAverage and displayMovingAverage?mafast:na, color=color.green, title="MAFast")
plot(considerMovingAverage and displayMovingAverage?maslow:na, color=color.red, title="MASlow")

barSinceBullishCrossover = barssince(crossover(mafast, maslow))
barSinceBearishCrossover = barssince(crossunder(mafast, maslow))


atr = atr(atrLength)
dbLowDiffs = lastLow - llastLow
dbHighLowDiff = lastHigh - (lastLow+llastLow)/2

dtHighDiffs = llastHigh - lastHigh
dtHighLowDiff = (llastHigh+lastHigh)/2 - lastLow

AtrL1 = atr[lastLowBar]
AtrL2 = atr[llastLowBar]
AtrH1 = atr[lastHighBar]
AtrH2 = atr[llastHighBar]

dbAtrBase = (AtrL1+AtrL2)/2
dbAtrHighLow = (AtrL1+AtrL2+AtrH1)/3

dtAtrBase = (AtrH1+AtrH2)/2
dtAtrHighLow = (AtrH1+AtrH2+AtrL1)/3

dtHighLowDiffInLimit = ((dtHighDiffs < dtAtrBase*maxAtrDistanceBase and dtHighLowDiff < dtAtrHighLow*m
dbHighLowDiffInLimit = ((dbLowDiffs < dbAtrBase*maxAtrDistanceBase and dbHighLowDiff < dbAtrHighLow*ma

doubleBottom = (crossover(highest(lastLowBar), lastHigh) or crossover(highest(close,lastLowBar), lastH
                    and lastHigh < llastHigh and lastLow > llastLow
                    and llastLow == lowest(llastHighBar)
                    and (not checkForAbsolutePeaks or llastLow == lowest(llastLowBar+absolutePeakLoop
                    and high == highest(lastHighBar)
                    and lastHighBar > lastLowBar and lastHighBar < llastLowBar
                    and ((mafast > maslow and  barSinceBullishCrossover < llastLowBar) or not conside
                    and dbHighLowDiffInLimit

doubleBottomOccurance = false
for i=1 to lastLowBar
    doubleBottomOccurance := doubleBottomOccurance or doubleBottom[i]
doubleBottom := doubleBottom and not doubleBottomOccurance

doubleTop = (crossunder(lowest(lastHighBar), lastLow) or crossunder(lowest(close, lastHighBar), lastLo
                    and lastHigh < llastHigh and lastLow > llastLow
                    and llastHigh == highest(llastLowBar)
                    and (not checkForAbsolutePeaks or llastHigh == highest(llastHighBar+absolutePeakL
```

```
                    and low == lowest(lastLowBar)
                    and lastHighBar < lastLowBar and llastHighBar > lastLowBar
                    and ((mafast < maslow and barSinceBearishCrossover < llastHighBar) or not conside
                    and dtHighLowDiffInLimit

doubleTopOccurance = false
for i=1 to lastHighBar
    doubleTopOccurance := doubleTopOccurance or doubleTop[i]
doubleTop := doubleTop and not doubleTopOccurance

if(barstate.islast and showFilteredPivots)
    for i = 0 to maxPivotArraySize-1
        label.new(array.get(ph_positions, i)-pvtLenR-Shunt, high, text=array.get(ph_highlow, i)? "HH":
                        color=array.get(ph_highlow, i) ? color.green : color.orange,
                        style=label.style_triangleup,
                        textcolor=color.white, size=size.auto)
        label.new(array.get(pl_positions, i)-pvtLenR-Shunt, low , text=array.get(pl_highlow, i)? "HL":
                        color=array.get(pl_highlow, i) ? color.lime: color.red,
                        style=label.style_triangledown,
                        textcolor=color.white, size=size.auto)


if(doubleTop)
    label.new(x=bar_index, y=high, text="Double Top", yloc=yloc.abovebar,
                                color=color.red,
                                style=label.style_label_down,
                                textcolor=color.black, size=size.large)
    line.new(x1=array.get(ph_positions, 0)-pvtLenR-Shunt, y1=array.get(ph_vals, 0),
                x2 = array.get(ph_positions, 1)-pvtLenR-Shunt, y2=array.get(ph_vals, 1),
                color=color.red, width=2)
    line.new(x1=array.get(pl_positions, 0)-pvtLenR-Shunt, y1=lastLow,
                x2 = bar_index, y2=lastLow,
                color=color.red, width=2, style=line.style_dashed)

if(doubleBottom)
    label.new(x=bar_index, y=low, text="Double Bottom", yloc=yloc.belowbar,
                                color=color.green,
                                style=label.style_label_up,
                                textcolor=color.black, size=size.large)
    line.new(x1=array.get(pl_positions, 0)-pvtLenR-Shunt, y1=array.get(pl_vals, 0),
                x2 = array.get(pl_positions, 1)-pvtLenR-Shunt, y2=array.get(pl_vals, 1),
                color=color.green, width=2)
    line.new(x1=array.get(ph_positions, 0)-pvtLenR-Shunt, y1=lastHigh,
                x2 = bar_index, y2=lastHigh,
                color=color.green, width=2, style=line.style_dashed)

barcolor(doubleBottom? color.lime : doubleTop? color.orange : colorCandles ? color.silver : na)
alertcondition(doubleTop, "Double Top", "Probable double top observed for {{ticker}} on {{interval}} t
alertcondition(doubleBottom, "Double Bottom", "Probable double bottom observed for {{ticker}} on {{int
```
